Package 'pliman'

June 10, 2021

Title Tools for Plant Image Analysis

Version 0.3.0

Description Provides tools for image manipulation that will help you to quantify plant leaf area, disease severity, number of disease lesions, and obtain statistics of image objects such as grains, pods, pollen, leaves, and more. Tools for segment images and create binary images using the method of automatic threshold selection method proposed by Otsu (1979) <doi:10.1109/tsmc.1979.4310076> are also provided.

License GPL (>= 3)

URL https://github.com/TiagoOlivoto/pliman

BugReports https://github.com/TiagoOlivoto/pliman/issues

Depends R (>= 4.1)

Imports EBImage, ggplot2

Suggests knitr, rmarkdown

VignetteBuilder knitr

biocViews

Encoding UTF-8

Language en-US

RoxygenNote 7.1.1

NeedsCompilation no

Author Tiago Olivoto [aut, cre] (<https://orcid.org/0000-0002-0241-9636>)

Maintainer Tiago Olivoto <tiagoolivoto@gmail.com>

Repository CRAN

Date/Publication 2021-06-10 04:50:02 UTC

R topics documented:

count_lesions . . . 2
count_objects . . . 5
image_binary . . . 9
image_combine . . . 11
image_index . . . 11
image_palette . . . 14
image_segment . . . 15
image_to_mat . . . 17
leaf_area . . . 18
objects_rgb . . . 20
pliman_images . . . 24
prop_segmented . . . 25
symptomatic_area . . . 26
utils_dpi . . . 28
utils_file . . . 29
utils_image . . . 31
utils_measures . . . 32
utils_objects . . . 34
utils_transform . . . 36

Index **39**

`count_lesions` *Counts the number of lesions*

Description

Counts the number of lesions in a sample or entire leaf based on provided color palettes samples. A general linear model (binomial family) fitted to the RGB values is used to segment the lesions from the healthy leaf. If a pallet of background is provided, the function takes care of the details to isolate it before computing the number and area of lesions. By using `img_pattern` it is possible to process several images with common pattern names that are stored in the current working directory or in the subdirectory informed in `dir_original`.

Usage

```
count_lesions(
  img,
  img_healthy = NULL,
  img_lesion = NULL,
  img_background = NULL,
  img_pattern = NULL,
  parallel = FALSE,
  workers = NULL,
  resize = FALSE,
  invert = FALSE,
```

```
  index = "NB",
  my_index = NULL,
  lower_size = NULL,
  upper_size = NULL,
  randomize = TRUE,
  nrows = 10000,
  lesion_size = "medium",
  segment = TRUE,
  tolerance = NULL,
  extension = NULL,
  show_segmentation = TRUE,
  show_image = FALSE,
  show_original = TRUE,
  show_background = TRUE,
  col_leaf = NULL,
  col_lesions = NULL,
  col_background = NULL,
  marker = NULL,
  marker_col = NULL,
  marker_size = NULL,
  save_image = FALSE,
  prefix = "proc_",
  dir_original = NULL,
  dir_processed = NULL,
  verbose = TRUE
)
```

Arguments

`img`	The image to be analyzed.
`img_healthy`	A color palette of healthy areas.
`img_lesion`	A color palette of lesioned areas.
`img_background`	An optional color palette of the image background.
`img_pattern`	A pattern of file name used to identify images to be processed. For example, if `img_pattern = "im"` all images that the name matches the pattern (e.g., img1.-, image1.-, im2.-) will be analyzed. Providing any number as pattern (e.g., `img_pattern = "1"`) will select images that are named as 1.-, 2.-, and so on.
`parallel`	Processes the images asynchronously (in parallel) in separate R sessions running in the background on the same machine. It may speed up the processing time, especially when `img_pattern` is used is informed. The number of sections is set up to 90% of available cores.
`workers`	A positive numeric scalar or a function specifying the maximum number of parallel processes that can be active at the same time.
`resize`	Resize the image before processing? Defaults to `FALSE`. Use a numeric value of range 0-100 (proportion of the size of the original image).
`invert`	Inverts the binary image, if desired. This is useful to process images with black background. Defaults to `FALSE`.

`index, my_index`
: A character value specifying the target mode for conversion to binary image when `img_healthy` and `img_lesion` are not declared. Defaults to `"NB"` (normalized blue). See `image_index()` for more details.

`lower_size`
: Lower limit for size for the image analysis. Leaf images often contain dirt and dust. To prevent dust from affecting the image analysis, the lower limit of analyzed size is set to 0.1, i.e., objects with lesser than 10% of the mean of all objects are removed. One can set a known area or use `lower_limit = 0` to select all objects (not advised).

`upper_size`
: Upper limit for size for the image analysis. Defaults to `NULL`, i.e., no upper limit used.

`randomize`
: Randomize the lines before training the model?

`nrows`
: The number of lines to be used in training step.

`lesion_size`
: The size of the lesion. Used to automatically set up `tolerance` and `extension` parameters. One of the following. `"small"` (2-5 mm in diameter, e.g, rust pustules), `"medium"` (0.5-1.0 cm in diameter, e.g, wheat leaf spot), `"large"` (1-2 cm in diameter, and `"elarge"` (2-3 cm in diameter, e.g, target spot of soybean).

`segment`
: If `TRUE` (Default) implements the Watershed Algorithm to segment lesions connected by a fairly few pixels that could be considered as two distinct lesions. If `FALSE`, lesions that are connected by any pixel are considered unique lesions. For more details see `EBImage::watershed()`.

`tolerance`
: The minimum height of the object in the units of image intensity between its highest point (seed) and the point where it contacts another object (checked for every contact pixel). If the height is smaller than the tolerance, the object will be combined with one of its neighbors, which is the highest. Defaults to `NULL`, i.e., starting values are set up according to the argument `lesion_size`.

`extension`
: Radius of the neighborhood in pixels for the detection of neighboring objects. Defaults to 20. Higher value smooths out small objects.

`show_segmentation`
: Shows the object segmentation colored with random permutations. Defaults to `TRUE`.

`show_image`
: Show image after processing?

`show_original`
: Show the symptoms in the original image?

`show_background`
: Show the background? Defaults to `TRUE`. A white background is shown by default when `show_original = FALSE`.

`col_leaf`
: Leaf color after image processing. Defaults to `"green"`

`col_lesions`
: Symptoms color after image processing. Defaults to `"red"`.

`col_background`
: Background color after image processing. Defaults to `"NULL"`.

`marker, marker_col, marker_size`
: The type, color and size of the object marker. Defaults to `NULL`, which shows a red point when `show_segmentation = FALSE`. To force a marker to be used with segmented objects, set up to `marker = "point"` (to show a point) or `marker = "text"` to enumerate the objects.

save_image Save the image after processing? The image is saved in the current working directory named as proc_* where * is the image name given in img.

prefix The prefix to be included in the processed images. Defaults to "proc_".

dir_original, dir_processed
The directory containing the original and processed images. Defaults to NULL. In this case, the function will search for the image img in the current working directory. After processing, when save_image = TRUE, the processed image will be also saved in such a directory.

verbose If TRUE (default) a summary is shown in the console.

Value

A list with the following objects:

- results A data frame with the results (area, perimeter, radius) for object.
- statistics A data frame with the summary statistics for the image.
- count (If img_pattern is used), summarizing the count number for each image.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

Examples

```
library(pliman)
img <- image_import(image_pliman("sev_leaf_nb.jpg"))
healthy <- image_import(image_pliman("sev_healthy.jpg"))
lesions <- image_import(image_pliman("sev_sympt.jpg"))
image_combine(img, healthy, lesions, ncol = 3)
a <-
count_lesions(img = img,
              img_healthy = healthy,
              img_lesion = lesions,
              lesion_size = "elarge", # extra large lesions
              show_image = TRUE,
              show_segmentation = FALSE,
              marker = "text")
```

---

count_objects *Computes number of objects in an image*

Description

Counts the number of objects in an image. See more at details.

Usage

```
count_objects(
  img,
  foreground = NULL,
  background = NULL,
  img_pattern = NULL,
  parallel = FALSE,
  workers = NULL,
  resize = FALSE,
  fill_hull = FALSE,
  filter = FALSE,
  invert = FALSE,
  index = "NB",
  my_index = NULL,
  object_size = "medium",
  tolerance = NULL,
  extension = NULL,
  lower_size = NULL,
  upper_size = NULL,
  topn_lower = NULL,
  topn_upper = NULL,
  randomize = TRUE,
  nrows = 10000,
  show_image = TRUE,
  show_original = TRUE,
  show_background = TRUE,
  show_segmentation = TRUE,
  col_foreground = NULL,
  col_background = NULL,
  marker = NULL,
  marker_col = NULL,
  marker_size = NULL,
  save_image = FALSE,
  prefix = "proc_",
  dir_original = NULL,
  dir_processed = NULL,
  verbose = TRUE
)
```

Arguments

`img`	The image to be analyzed.
`foreground`	A color palette of the foreground (optional).
`background`	A color palette of the background (optional).
`img_pattern`	A pattern of file name used to identify images to be processed. For example, if `img_pattern = "im"` all images that the name matches the pattern (e.g., img1.-, image1.-, im2.-) will be analyzed. Providing any number as pattern (e.g., `img_pattern = "1"`) will select images that are named as 1.-, 2.-, and so on.

`parallel` Processes the images asynchronously (in parallel) in separate R sessions running in the background on the same machine. It may speed up the processing time, especially when `img_pattern` is used is informed. The number of sections is set up to 90% of available cores.

`workers` A positive numeric scalar or a function specifying the maximum number of parallel processes that can be active at the same time.

`resize` Resize the image before processing? Defaults to `FALSE`. Use a numeric value of range 0-100 (proportion of the size of the original image).

`fill_hull` Fill holes in the binary image? Defaults to `FALSE`. This is useful to fill holes in objects that have portions with a color similar to the background. IMPORTANT: Objects touching each other can be combined into one single object, which may underestimate the number of objects in an image.

`filter` Performs median filtering after image processing? defaults to `FALSE`. See more at `image_filter()`.

`invert` Inverts the binary image, if desired. This is useful to process images with black background. Defaults to `FALSE`.

`index, my_index`
A character value specifying the target mode for conversion to binary image when `foreground` and `background` are not declared. Defaults to `"NB"` (normalized blue). See `image_index()` for more details.

`object_size` The size of the object. Used to automatically set up `tolerance` and `extension` parameters. One of the following. `"small"` (e.g, wheat grains), `"medium"` (e.g, soybean grains), `"large"`(e.g, peanut grains), and `"elarge"` (e.g, soybean pods)‘.

`tolerance` The minimum height of the object in the units of image intensity between its highest point (seed) and the point where it contacts another object (checked for every contact pixel). If the height is smaller than the tolerance, the object will be combined with one of its neighbors, which is the highest.

`extension` Radius of the neighborhood in pixels for the detection of neighboring objects. Defaults to 20. Higher value smooths out small objects.

`lower_size, upper_size`
Lower and upper limits for size for the image analysis. Plant images often contain dirt and dust. To prevent dust from affecting the image analysis, objects with lesser than 10% of the mean of all objects are removed. Upper limit is set to `NULL`, i.e., no upper limit used. One can set a known area or use `lower_limit = 0` to select all objects (not advised). Objects that matches the size of a given range of sizes can be selected by setting up the two arguments. For example, if `lower_size = 120` and `upper_size = 140`, objects with size greater than or equal 120 and less than or equal 140 will be considered.

`topn_lower, topn_upper`
Select the top `n` objects based on its area. `topn_lower` selects the `n` elements with the smallest area whereas `topn_upper` selects the `n` objects with the largest area.

`randomize` Randomize the lines before training the model?

`nrows` The number of lines to be used in training step.

show_image Show image after processing?

show_original Show the count objects in the original image?

show_background
Show the background? Defaults to TRUE. A white background is shown by default when show_original = FALSE.

show_segmentation
Shows the object segmentation colored with random permutations. Defaults to TRUE.

col_foreground, col_background
Foreground and background color after image processing. Defaults to NULL, in which "black", and "white" are used, respectively.

marker, marker_col, marker_size
The type, color and size of the object marker. Defaults to NULL, which shows a red point when show_segmentation = FALSE. To force a marker to be used with segmented objects, set up to marker = "point" (to show a point) or marker = "text" to enumerate the objects.

save_image Save the image after processing? The image is saved in the current working directory named as proc_* where * is the image name given in img.

prefix The prefix to be included in the processed images. Defaults to "proc_".

dir_original, dir_processed
The directory containing the original and processed images. Defaults to NULL. In this case, the function will search for the image img in the current working directory. After processing, when save_image = TRUE, the processed image will be also saved in such a directory.

verbose If TRUE (default) a summary is shown in the console.

Details

Counts the number of objects in an image. A binary image is first generated to segment the foreground and background. The argument index is useful to choose a proper index to segment the image (see image_binary() for more details). Then, the number of objects in the foreground is counted. By setting up arguments such as lower_size, upper_size is possible to set a threshold for lower and upper sizes of the objects, respectively. Change tolerance and extension values to better set up watershed-based object detection. If color palettes samples are provided, a general linear model (binomial family) fitted to the RGB values is used to segment fore- and background.

By using img_pattern it is possible to process several images with common pattern names that are stored in the current working directory or in the subdirectory informed in dir_original'. To speed up the computation time, one can set parallel = TRUE.

Value

A list with the following objects:

- results A data frame with the results (area, perimeter, radius) for object.
- statistics A data frame with the summary statistics for the image.
- count (If img_pattern is used), summarizing the count number for each image.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

Examples

```
library(pliman)
img <- image_import(image_pliman("soybean_touch.jpg"))
count_objects(img)

# Enumerate the objects in the original image
count_objects(img,
              show_segmentation = FALSE,
              marker = "text",
              marker_col = "white")
```

---

image_binary *Creates a binary image*

Description

Reduce a color, color near-infrared, or grayscale images to a binary image using a given color channel (red, green blue) or even color indexes. The Otsu's thresholding method (Otsu, 1979) is used to automatically perform clustering-based image thresholding.

Usage

```
image_binary(
  image,
  index = NULL,
  my_index = NULL,
  resize = 30,
  fill_hull = FALSE,
  re = NULL,
  nir = NULL,
  invert = FALSE,
  show_image = TRUE,
  nrow = NULL,
  ncol = NULL,
  parallel = FALSE,
  workers = NULL,
  verbose = TRUE
)
```

Arguments

image	An image object.
index	A character value (or a vector of characters) specifying the target mode for conversion to binary image. One of the following: "R", "G", "B" "GR", "NR", "NG", "NB", "BI", "BIM", "SCI", "GLI", "HI", "NGRDI", "SI", "VARI", "HUE", "HUE2", "BGI", "BGI". See image_index() for more details.
my_index	User can calculate a different index using the bands names, e.g. my_index = "R+B/G".
resize	Resize the image before processing? Defaults to 30, which resizes the image to 30% of the original size to speed up image processing. Set resize = FALSE to keep the original size of the image.
fill_hull	Fill holes in the objects? Defaults to FALSE.
re	Respective position of the red-edge band at the original image file.
nir	Respective position of the near-infrared band at the original image file.
invert	Inverts the binary image, if desired.
show_image	Show image after processing?
nrow, ncol	The number of rows or columns in the plot grid. Defaults to NULL, i.e., a square grid is produced.
parallel	Processes the images asynchronously (in parallel) in separate R sessions running in the background on the same machine. It may speed up the processing time when image is a list. The number of sections is set up to 90% of available cores.
workers	A positive numeric scalar or a function specifying the maximum number of parallel processes that can be active at the same time.
verbose	If TRUE (default) a summary is shown in the console.

Value

A list containing binary images. The length will depend on the number of indexes used.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

References

Nobuyuki Otsu, "A threshold selection method from gray-level histograms". IEEE Trans. Sys., Man., Cyber. 9 (1): 62-66. 1979. doi: 10.1109/TSMC.1979.4310076

Examples

```
library(pliman)
img <- image_import(image_pliman("soybean_touch.jpg"))
image_binary(img, index = c("R, NR"), nrow = 1)
```

image_combine *Combines images to a grid*

Description

Combines several images to a grid

Usage

```
image_combine(..., nrow = NULL, ncol = NULL)
```

Arguments

...	a comma-separated name of image objects or a list containing image objects.
nrow, ncol	The number of rows or columns in the plot grid. Defaults to NULL, i.e., a square grid is produced.

Value

A grid with the images in ...

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

Examples

```
library(pliman)
img1 <- image_import(image_pliman("sev_leaf.jpg"))
img2 <- image_import(image_pliman("sev_leaf_nb.jpg"))
image_combine(img1, img2)
```

image_index *Image indexes*

Description

Builds image indexes using Red, Green, Blue, Red-Edge, and NIR bands.

Produces an histogram of an image_index object

Usage

```
image_index(
  image,
  index = NULL,
  my_index = NULL,
  resize = 30,
  re = NULL,
  nir = NULL,
  show_image = TRUE,
  nrow = NULL,
  ncol = NULL,
  parallel = FALSE,
  workers = NULL,
  verbose = TRUE
)

## S3 method for class 'image_index'
plot(x, facet = TRUE, ...)
```

Arguments

image	An image object.
index	A character value (or a vector of characters) specifying the target mode for conversion to binary image. One of the following: "R", "G", "B" "GR", "NR", "NG", "NB", "BI", "BIM", "SCI", "GLI", "HI", "NGRDI", "SI", "VARI", "HUE", "HUE2", "BGI", "BGI". Defaults to NULL ((normalized) Red, Green and Blue). One can also use "RGB" for RGB only, "NRGB" for normalized RGB, or "all" for all indexes.
my_index	User can calculate a different index using the bands names, e.g. my_index = "R+B/G".
resize	Resize the image before processing? Defaults to 30, which resizes the image to 30% of the original size to speed up image processing. Set resize = FALSE to keep the original size of the image.
re	Respective position of the red-edge band at the original image file.
nir	Respective position of the near-infrared band at the original image file.
show_image	Show image after processing?
nrow, ncol	The number of rows or columns in the plot grid. Defaults to NULL, i.e., a square grid is produced.
parallel	Processes the images asynchronously (in parallel) in separate R sessions running in the background on the same machine. It may speed up the processing time when image is a list. The number of sections is set up to 90% of available cores.
workers	A positive numeric scalar or a function specifying the maximum number of parallel processes that can be active at the same time.
verbose	If TRUE (default) a summary is shown in the console.
x	An object of class image_index.

facet	Shows RGB values as a facet plot? Defaults to TRUE.
...	Currently not used

Details

The following indexes are available in pliman.

Index	Equation	Band
R	R	C
G	G	C
B	B	C
NR	R/(R+G+B)	C
NG	G/(R+G+B)	C
NB	B/(R+G+B)	C
GB	G/B	C
RB	R/B	C
GR	G/R	C
BI	sqrt((R^2+G^2+B^2)/3)	C
BIM	sqrt((R*2*+G*2+B*2)/3)	C
SCI	(R-G)/(R+G)	C
GLI	(*2G-R-B)/(2*G+R+B)	C
HI	(2*R-G-B)/(G-B)	C
NDGRI	(G-R)/(G+R)	C
NDGBI	(G-B)/(G+B)	C
NDRBI	(R-B)/(R+B)	C
I	R+G+B	C
S	((R+G+B)-3*B)/(R+G+B)	C
L	R+G+B/3	C
VARI	(G-R)/(G+R-B)	C
HUE	atan(2*(B-G-R)/30.5*(G-R))	C
HUE2	atan(2*(R-G-R)/30.5*(G-B))	C
BGI	B/G	C

Value

A list containing Grayscale images. The length will depend on the number of indexes used.

A ggplot object containing the distribution of the pixels for each index.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

Tiago Olivoto <tiagoolivoto@gmail.com>

References

Nobuyuki Otsu, "A threshold selection method from gray-level histograms". IEEE Trans. Sys., Man., Cyber. 9 (1): 62-66. 1979. doi: 10.1109/TSMC.1979.4310076

Examples

```
library(pliman)
img <- image_import(image_pliman("soybean_touch.jpg"))
image_index(img, c("R, NR"), nrow = 1)
library(pliman)
img <- image_import(image_pliman("sev_leaf.jpg"))
img2 <- image_resize(img, 50)
ind <- image_index(img2)
plot(ind)
```

---

image_palette *Create an image palette*

Description

Creates image palettes by applying the k-means algorithm to the RGB values.

Usage

```
image_palette(
  image,
  npal,
  nstart = 25,
  parallel = FALSE,
  workers = NULL,
  verbose = TRUE
)
```

Arguments

image	An image object.
npal	The number of color palettes.
nstart	How many random sets from npal should be chosen?
parallel	Processes the images asynchronously (in parallel) in separate R sessions running in the background on the same machine. It may speed up the processing time when image is a list. The number of sections is set up to 90% of available cores.
workers	A positive numeric scalar or a function specifying the maximum number of parallel processes that can be active at the same time.
verbose	If TRUE (default) a summary is shown in the console.

Value

A list with npal color palettes of class Image.

Examples

```
library(pliman)
img <- image_import(image_pliman("sev_leaf.jpg"))
pal <- image_palette(img, 2)
image_show(pal[[1]])
image_show(pal[[2]])
```

`image_segment` *Image segmentation*

Description

Reduce a color, color near-infrared, or grayscale images to a segmented image using a given color channel (red, green blue) or even color indexes (See `image_index()` for more details). The Otsu's thresholding method (Otsu, 1979) is used to automatically perform clustering-based image thresholding.

Usage

```
image_segment(
  image,
  index = NULL,
  my_index = NULL,
  fill_hull = FALSE,
  re = NULL,
  nir = NULL,
  invert = FALSE,
  show_image = TRUE,
  nrow = NULL,
  ncol = NULL,
  parallel = FALSE,
  workers = NULL,
  verbose = TRUE
)
```

Arguments

`image`	An image object.
`index`	A character value (or a vector of characters) specifying the target mode for conversion to binary image. One of the following: `"R"`, `"G"`, `"B"` `"GR"`, `"NR"`, `"NG"`, `"NB"`, `"BI"`, `"BIM"`, `"SCI"`, `"GLI"`, `"HI"`, `"NGRDI"`, `"SI"`, `"VARI"`, `"HUE"`, `"HUE2"`, `"BGI"`, `"BGI"`. See `image_index()` for more details.
`my_index`	User can calculate a different index using the bands names, e.g. `my_index = "R+B/G"`.

fill_hull	Fill holes in the objects? Defaults to FALSE.
re	Respective position of the red-edge band at the original image file.
nir	Respective position of the near-infrared band at the original image file.
invert	Inverts the binary image, if desired.
show_image	Show image after processing?
nrow, ncol	The number of rows or columns in the plot grid. Defaults to NULL, i.e., a square grid is produced.
parallel	Processes the images asynchronously (in parallel) in separate R sessions running in the background on the same machine. It may speed up the processing time when image is a list. The number of sections is set up to 90% of available cores.
workers	A positive numeric scalar or a function specifying the maximum number of parallel processes that can be active at the same time.
verbose	If TRUE (default) a summary is shown in the console.

Value

A list containing n objects where n is the number of indexes used. Each objects contains:

- image an image with the RGB bands (layers) for the segmented object.
- mask A mask with logical values of 0 and 1 for the segmented image.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

References

Nobuyuki Otsu, "A threshold selection method from gray-level histograms". IEEE Trans. Sys., Man., Cyber. 9 (1): 62-66. 1979. doi: 10.1109/TSMC.1979.4310076

Examples

```
library(pliman)
img <- image_import(image_pliman("soybean_touch.jpg"))
image_show(img)
image_segment(img, index = c("R, G, B"))
```

image_to_mat *Convert an image to numerical matrices*

Description

Given an object image, converts it into three matrices (RGB) and a data frame where each column corresponds to the RGB values.

Usage

```
image_to_mat(image, parallel = FALSE, workers = NULL, verbose = TRUE)
```

Arguments

image	An image object.
parallel	Processes the images asynchronously (in parallel) in separate R sessions running in the background on the same machine. It may speed up the processing time when image is a list. The number of sections is set up to 90% of available cores.
workers	A positive numeric scalar or a function specifying the maximum number of parallel processes that can be active at the same time.
verbose	If TRUE (default) a summary is shown in the console.

Value

A list containing three matrices (R, G, and B), and a data frame containing four columns: the name of the image in image and the R, G, B values.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

Examples

```
library(pliman)
img <- image_import(image_pliman("sev_leaf.jpg"))
dim(img)
mat <- image_to_mat(img)
dim(mat[[1]])
```

leaf_area *Calculates the leaf area*

Description

- `leaf_area()` Calculates the leaf area using an image with leaves and a template with a known area. A general linear model (binomial family) fitted to the RGB values is used to first separate the leaves and template from the background and then the leaves from the template. The leaf area is then calculated for each leaf based on the pixel area. By using `img_pattern` it is possible to process several images with common pattern names that are stored in the current working directory or in the subdirectory informed in `dir_originals`.

Usage

```
leaf_area(
  img,
  img_leaf,
  img_background,
  img_template,
  area_template,
  resize = FALSE,
  parallel = FALSE,
  workers = NULL,
  img_pattern = NULL,
  lower_size = NULL,
  upper_size = NULL,
  randomize = TRUE,
  nrows = 10000,
  show_image = TRUE,
  show_original = TRUE,
  show_background = TRUE,
  col_background = NULL,
  col_leaf = "green",
  text_col = "black",
  text_size = 1,
  text_digits = 2,
  save_image = FALSE,
  dir_original = NULL,
  dir_processed = NULL,
  verbose = TRUE
)
```

Arguments

`img`	The image to be analyzed.
`img_leaf`	A color palette of the leaves.

`img_background` A color palette of background area.

`img_template` A color palette of the template areas.

`area_template` The known area of the template. The leaf area will be given in the same unit as `area_template`.

`resize` Resize the image before processing? Defaults to `FALSE`. Use a numeric value of range 0-100 (proportion of the size of the original image).

`parallel` Processes the images asynchronously (in parallel) in separate R sessions running in the background on the same machine. It may speed up the processing time, especially when `img_pattern` is used is informed. The number of sections is set up to 90% of available cores.

`workers` A positive numeric scalar or a function specifying the maximum number of parallel processes that can be active at the same time.

`img_pattern` A pattern of file name used to identify images to be processed. For example, if `img_pattern = "im"` all images that the name matches the pattern (e.g., img1.-, image1.-, im2.-) will be analyzed. Providing any number as pattern (e.g., `img_pattern = "1"`) will select images that are named as 1.-, 2.-, and so on.

`lower_size` Lower limit for size for the image analysis. Leaf images often contain dirt and dust. To prevent dust from affecting the image analysis, the lower limit of analyzed size is set to 0.1, i.e., objects with lesser than 10% of the mean of all objects are removed. One can set a known area or use `lower_limit = 0` to select all objects (not advised).

`upper_size` Upper limit for size for the image analysis. Defaults to `NULL`, i.e., no upper limit used.

`randomize` Randomize the lines before training the model?

`nrows` The number of lines to be used in training step.

`show_image` Show image after processing?

`show_original` Show the symptoms in the original image?

`show_background`
Show the background? Defaults to `TRUE`. A white background is shown by default when `show_original = FALSE`.

`col_background` Background color after image processing.

`col_leaf` Leaf color after image processing when `show_original = FALSE`. Defaults to `"green"`.

`text_col, text_size, text_digits`
The color, size and significant digits used in the text. The shows the pattern `o|a`, where `o` and `a` are the object id and its area, respectively.

`save_image` Save the image after processing? The image is saved in the current working directory named as proc_* where * is the image name given in `img`.

`dir_original, dir_processed`
The directory containing the original and processed images. Defaults to `NULL`. In this case, the function will search for the image `img` in the current working directory. After processing, when `save_image = TRUE`, the processed image will be also saved in such a directory.

`verbose` If `TRUE` (default) a summary is shown in the console.

Value

A data frame with the results for each image.

Examples

```
library(pliman)
img <- image_import(image_pliman("la_pattern.JPG"))
leaf <- image_import(image_pliman("la_leaf.jpg"))
tmpl <- image_import(image_pliman("la_temp.jpg"))
background <- image_import(image_pliman("la_back.jpg"))

# Combine the images
image_combine(img, leaf, tmpl, background)

# Computes the leaf area
area <-
leaf_area(img = img,
          img_leaf = leaf,
          img_template = tmpl,
          img_background = background,
          area_template = 4,
          text_col = "white")
get_measures(area)
```

---

objects_rgb *Get Red Green and Blue for image objects*

Description

Get the Red Green and Blue (RGB) for objects in an image.

Produces an histogram of an objects_rgb object

Usage

```
objects_rgb(
  img,
  foreground = NULL,
  background = NULL,
  img_pattern = NULL,
  parallel = FALSE,
  workers = NULL,
  resize = FALSE,
  fill_hull = FALSE,
  invert = FALSE,
  index = "NB",
```

```
  my_index = NULL,
  object_index = "B",
  object_size = "large",
  tolerance = NULL,
  extension = NULL,
  lower_size = NULL,
  upper_size = NULL,
  topn_lower = NULL,
  topn_upper = NULL,
  nrows = 10000,
  show_image = TRUE,
  save_image = FALSE,
  prefix = "proc_",
  marker = NULL,
  marker_col = NULL,
  marker_size = NULL,
  marker_digits = NULL,
  dir_original = NULL,
  dir_processed = NULL,
  verbose = TRUE
)

## S3 method for class 'objects_rgb'
plot(x, ...)
```

Arguments

`img`	The image to be analyzed.
`foreground`	A color palette of the foreground (optional).
`background`	A color palette of the background (optional).
`img_pattern`	A pattern of file name used to identify images to be processed. For example, if `img_pattern = "im"` all images that the name matches the pattern (e.g., img1.-, image1.-, im2.-) will be analyzed. Providing any number as pattern (e.g., `img_pattern = "1"`) will select images that are named as 1.-, 2.-, and so on.
`parallel`	Processes the images asynchronously (in parallel) in separate R sessions running in the background on the same machine. It may speed up the processing time, especially when `img_pattern` is used is informed. The number of sections is set up to 90% of available cores.
`workers`	A positive numeric scalar or a function specifying the maximum number of parallel processes that can be active at the same time.
`resize`	Resize the image before processing? Defaults to `FALSE`. Use a numeric value of range 0-100 (proportion of the size of the original image).
`fill_hull`	Fill holes in the binary image? Defaults to `FALSE`. This is useful to fill holes in objects that have portions with a color similar to the background. IMPORTANT: Objects touching each other can be combined into one single object, which may underestimate the number of objects in an image.

`invert` Inverts the binary image, if desired. This is useful to process images with black background. Defaults to `FALSE`.

`index, my_index`
A character value specifying the target mode for conversion to binary image when `foreground` and `background` are not declared. Defaults to `"NB"` (normalized blue). See `image_index()` for more details.

`object_index` The same as `index`, used to compute the index for each object in the image.

`object_size` The size of the object. Used to automatically set up `tolerance` and `extension` parameters. One of the following. `"small"` (e.g, wheat grains), `"medium"` (e.g, soybean grains), `"large"`(e.g, peanut grains), and `"elarge"` (e.g, soybean pods)‘.

`tolerance` The minimum height of the object in the units of image intensity between its highest point (seed) and the point where it contacts another object (checked for every contact pixel). If the height is smaller than the tolerance, the object will be combined with one of its neighbors, which is the highest.

`extension` Radius of the neighborhood in pixels for the detection of neighboring objects. Defaults to 20. Higher value smooths out small objects.

`lower_size, upper_size`
Lower and upper limits for size for the image analysis. Plant images often contain dirt and dust. To prevent dust from affecting the image analysis, objects with lesser than 10% of the mean of all objects are removed. Upper limit is set to `NULL`, i.e., no upper limit used. One can set a known area or use `lower_limit = 0` to select all objects (not advised). Objects that matches the size of a given range of sizes can be selected by setting up the two arguments. For example, if `lower_size = 120` and `upper_size = 140`, objects with size greater than or equal 120 and less than or equal 140 will be considered.

`topn_lower, topn_upper`
Select the top `n` objects based on its area. `topn_lower` selects the `n` elements with the smallest area whereas `topn_upper` selects the `n` objects with the largest area.

`nrows` The number of lines to be used in training step.

`show_image` Show image after processing? Defaults to `TRUE`.

`save_image` Save the image after processing? The image is saved in the current working directory named as proc_* where * is the image name given in `img`.

`prefix` The prefix to be included in the processed images. Defaults to `"proc_"`.

`marker, marker_col, marker_size, marker_digits`
The marker, color, size and significant digits of the object marker. Defaults to `marker = "index"`, which shows the object index. Set to `marker = "id"` to show the object id.

`dir_original, dir_processed`
The directory containing the original and processed images. Defaults to `NULL`. In this case, the function will search for the image `img` in the current working directory. After processing, when `save_image = TRUE`, the processed image will be also saved in such a directory.

`verbose` If `FALSE`, runs the code silently.

x	An object of class objects_rgb.
...	Currently not used

Details

A binary image is first generated to segment the foreground and background. The argument index is useful to choose a proper index to segment the image (see image_binary() for more details). Then, the number of objects in the foreground is counted. Change tolerance and extension values to better set up watershed-based object detection. If color palettes samples are provided, a general linear model (binomial family) fitted to the RGB values is used to segment fore- and background. For each segmented object, the RGB values are obtained with. Users can also compute an index for each object using the argument object_index, useful to classify objects based on its RGB values.

By using img_pattern it is possible to process several images with common pattern names that are stored in the current working directory or in the subdirectory informed in dir_original'. To speed up the computation time, one can set parallel = TRUE.

Value

A list with the following objects.

- objects A data frame with the measures for each object.
- rgb A data frame with the Red, Green and Blue values for each object
- indexes A data frame with the index computed according to the argument object_index.

A ggplot object containing the distribution of the pixels for each index.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

Tiago Olivoto <tiagoolivoto@gmail.com>

Examples

```
library(pliman)
img <- image_import(image_pliman("soy_green.jpg"))
# Segment the foreground (grains) using the normalized blue index
# Shows the average value of the blue index in each object
rgb <- objects_rgb(img)
plot(rgb)
```

```
library(pliman)

img <- image_import(image_pliman("soy_green.jpg"))
# Segment the foreground (grains) using the normalized blue index
# Shows the average value of the blue index in each object
rgb <- objects_rgb(img)
plot(rgb)
```

`pliman_images` *Sample images*

Description

Sample images installed with the **pliman** package

Format

*.jpg format

- `la_back.jpg` A cyan palette representing the background of images la_pattern, la_leaves, and soybean_touch.
- `la_leaf.jpg` A sample of the leaves in `la_leaves`
- `la_leaves.JPG` Tree leaves with a sample of known area.
- `la_pattern.JPG` Tree leaves with a yellow sample of known area.
- `la_temp.jpg` The yellow sample of `la_pattern.JPG`.
- `objects_300dpi.jpg` An image with 300 dpi resolution.
- `sev_leaf.jpg` A soybean leaf with a blue background.
- `sev_leaf_nb.jpg` A soybean leaf without background.
- `sev_back.jpg` A blue palette representing the background of `sev_leaf`.
- `sev_healthy.jpg` Healthy area of `sev_leaf`.
- `sev_sympt.jpg` The symptomatic area `sev_leaf`.
- `soy_green.jpg` Soybean grains with a white background.
- `soybean_grain.jpg` A sample palette of the grains in `soy_green`.
- `soybean_touch.jpg` Soybean grains with a cyan background touching one each other.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

Source

Personal data

prop_segmented *Image segmentation with pixels proportion*

Description

Provides (iterative) image segmentation and get the proportion of pixels. This can be used as an alternative way to compute disease severity without using color palettes.

Usage

```
prop_segmented(
  image,
  nseg = 1,
  index = NULL,
  fill_hull = FALSE,
  filter = FALSE,
  parallel = FALSE,
  workers = NULL,
  show_image = TRUE,
  ncol = NULL,
  nrow = NULL,
  verbose = TRUE
)
```

Arguments

image	An object (or a list of objects) of class Image.
nseg	The number of iterative segmentation steps to be performed.
index	The index to be used in the image segmentation. See image_index() for more details.
fill_hull	Fill holes in the objects? Defaults to FALSE.
filter	Performs median filtering after image processing? defaults to FALSE. See more at image_filter().
parallel	Processes the images asynchronously (in parallel) in separate R sessions running in the background on the same machine. It may speed up the processing time when image is a list. The number of sections is set up to 90% of available cores.
workers	A positive numeric scalar or a function specifying the maximum number of parallel processes that can be active at the same time.
show_image	Show the image results? Defaults to TRUE.
nrow, ncol	Arguments passed on to image_combine(). The number of rows or columns in the plot grid. Defaults to NULL, i.e., a square grid is produced.
verbose	If TRUE (default) a summary is shown in the console.

Value

A list with the following objects

- `results` A data frame with the number of pixels and proportion of pixels in relation to the previous segmentation.
- `images` A list of segmented images.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

Examples

```
img <- image_import(image_pliman("sev_leaf.jpg"))
image_show(img)

prop_segmented(img,
               nseg = 2,
               index = c("G", "GLI"),
               ncol = 3)
```

`symptomatic_area` *Calculates the percentage of symptomatic area*

Description

Calculates the percentage of symptomatic leaf area in sample or entire leaf based on provided color palettes samples. A general linear model (binomial family) fitted to the RGB values is used to segment the lesions from the healthy leaf. If a pallet of background is provided, the function takes care of the details to isolate it before computing the number and area of lesions. By using `img_pattern` it is possible to process several images with common pattern names that are stored in the current working directory or in the subdirectory informed in `dir_original`.

Usage

```
symptomatic_area(
  img,
  img_healthy,
  img_symptoms,
  img_background = NULL,
  img_pattern = NULL,
  resize = FALSE,
  parallel = FALSE,
  workers = NULL,
  randomize = TRUE,
  nrows = 10000,
```

```
  show_image = FALSE,
  show_original = TRUE,
  show_background = TRUE,
  col_leaf = "green",
  col_symptoms = "red",
  col_background = NULL,
  save_image = FALSE,
  prefix = "proc_",
  dir_original = NULL,
  dir_processed = NULL,
  verbose = TRUE
)
```

Arguments

`img`	The image to be analyzed.
`img_healthy`	A color palette of healthy areas.
`img_symptoms`	A color palette of symptomatic areas.
`img_background`	A color palette of areas with symptoms.
`img_pattern`	A pattern of file name used to identify images to be processed. For example, if `img_pattern = "im"` all images that the name matches the pattern (e.g., img1.-, image1.-, im2.-) will be analyzed. Providing any number as pattern (e.g., `img_pattern = "1"`) will select images that are named as 1.-, 2.-, and so on.
`resize`	Resize the image before processing? Defaults to `FALSE`. Use a numeric value of range 0-100 (proportion of the size of the original image).
`parallel`	Processes the images asynchronously (in parallel) in separate R sessions running in the background on the same machine. It may speed up the processing time, especially when `img_pattern` is used is informed. The number of sections is set up to 90% of available cores.
`workers`	A positive numeric scalar or a function specifying the maximum number of parallel processes that can be active at the same time.
`randomize`	Randomize the lines before training the model?
`nrows`	The number of lines to be used in training step.
`show_image`	Show image after processing?
`show_original`	Show the symptoms in the original image?
`show_background`	Show the background? Defaults to `TRUE`. A white background is shown by default when `show_original = FALSE`.
`col_leaf`	Leaf color after image processing. Defaults to `"green"`
`col_symptoms`	Symptoms color after image processing. Defaults to `"red"`.
`col_background`	Background color after image processing. Defaults to `"NULL"`.
`save_image`	Save the image after processing? The image is saved in the current working directory named with the prefix provided in proc_* where * is the image name given in `img`.

prefix The prefix to be included in the processed images. Defaults to "proc_".

dir_original, dir_processed
The directory containing the original and processed images. Defaults to NULL. In this case, the function will search for the image img in the current working directory. After processing, when save_image = TRUE, the processed image will be also saved in such a directory.

verbose If TRUE (default) a summary is shown in the console.

Value

A data frame with the results (healthy and symptomatic area) for each image.

Examples

```
library(pliman)
img <- image_import(image_pliman("sev_leaf.jpg"))
healthy <- image_import(image_pliman("sev_healthy.jpg"))
symptoms <- image_import(image_pliman("sev_sympt.jpg"))
background <- image_import(image_pliman("sev_back.jpg"))
image_combine(img, healthy, symptoms,background)
symptomatic_area(img = img,
                 img_healthy = healthy,
                 img_symptoms = symptoms,
                 img_background = background,
                 show_image = TRUE)
```

---

utils_dpi *Utilities for image resolution*

Description

Provides useful conversions between size (cm), number of pixels (px) and resolution (dpi)

Usage

```
dpi_to_cm(dpi)

cm_to_dpi(cm)

pixels_to_cm(px, dpi)

cm_to_pixels(cm, dpi)
```

Arguments

dpi	The image resolution in dots per inch.
cm	The size in centimeters.
px	The number of pixels.

Value

A numeric value or a vector of numeric values if the input data is a vector

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

Examples

```
library(pliman)
# Convert  dots per inch to centimeter
dpi_to_cm(c(1, 2, 3))

# Convert centimeters to dots per inch
cm_to_dpi(c(1, 2, 3))

# Convert centimeters to number of pixels with resolution of 96 dpi.
cm_to_pixels(c(1, 2, 3), 96)

# Convert number of pixels to cm with resolution of 96 dpi.
pixels_to_cm(c(1, 2, 3), 96)
```

---

utils_file *Utilities for file manipulation*

Description

- `file_extension()` Get the extension of a file.
- `file_name()` Get the name of a file.
- `file_dir()` Get or directory of a file
- `manipulate_files()` Manipulate files in a directory with options to rename (insert prefix or suffix) and save the new files to the same or other provided directory.
- `pliman_indexes()` Get the indexes available in pliman.

Usage

```
file_extension(file)

file_name(file)

file_dir(file)

manipulate_files(
  pattern,
  dir = NULL,
  prefix = NULL,
  name = NULL,
  suffix = NULL,
  sep = "",
  save_to = NULL,
  overwrite = FALSE,
  remove_original = FALSE,
  verbose = TRUE
)

pliman_indexes()
```

Arguments

file	The file name.
pattern	A file name pattern.
dir	The working directory containing the files to be manipulated. Defaults to the current working directory.
prefix, suffix	A prefix or suffix to be added in the new file names. Defaults to NULL (no prefix or suffix).
name	The name of the new files. Defaults to NULL (original names). name can be either a single value or a character vector of the same length as the number of files manipulated. If one value is informed, a sequential vector of names will be created as "name_1", "name_2", and so on.
sep	An optional separator. Defaults to "".
save_to	The directory to save the new files. Defaults to the current working directory. If the file name of a file is not changed, nothing will occur. If save_to refers to a subfolder in the current working directory, the files will be saved to the given folder. In case of the folder doesn't exist, it will be created. By default, the files will not be overwritten. Set overwrite = TRUE to overwrite the files.
overwrite	Overwrite the files? Defaults to FALSE.
remove_original	Remove original files after manipulation? defaults to FALSE. If TRUE the files in pattern will be removed.
verbose	If FALSE, the code is run silently.

Value

- `file_extension()`, `file_name()`, and `file_dir()` return a character string.
- `manipulate_files()` No return value. If `verbose == TRUE`, a message is printed indicating which operation succeeded (or not) for each of the files attempted.

Examples

```
library(pliman)
# get file name, directory and extension
file <- "E:/my_folder/my_subfolder/image1.png"
file_dir(file)
file_name(file)
file_extension(file)

# manipulate files
dir <- tempdir()
list.files(dir)
file.create(paste0(dir, "/test.txt"))
list.files(dir)
manipulate_files("test",
                 dir = paste0(dir, "\\"),
                prefix = "chang_",
                save_to = paste0(dir, "\\"),
                overwrite = TRUE)
list.files(dir)
```

---

utils_image *Import, display and export images*

Description

Import images from files and URLs, write images to files, and show images. *

Usage

```
image_import(image, ..., img_pattern = NULL)

image_export(image, name, ...)

image_show(image)

image_pliman(image)
```

Arguments

image	• For image_import(), a character vector of file names or URLs. • For image_export(), an Image object, an array or a list of images.
...	Alternative arguments passed to the corresponding functions from the jpeg, png, and tiff packages.
img_pattern	A pattern of file name used to identify images to be imported. For example, if img_pattern = "im" all images in the current working directory that the name matches the pattern (e.g., img1.-, image1.-, im2.-) will be imported as a list. Providing any number as pattern (e.g., img_pattern = "1") will select images that are named as 1.-, 2.-, and so on.
name	An string specifying the name of the image.

Value

- image_import() returns a new Image object.
- image_export() returns an invisible vector of file names.
- image_pliman() returns a character string with the path to the example image installed with the package.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

Examples

```
library(pliman)
img <- image_import(image_pliman("sev_leaf.jpg"))
```

---

utils_measures *Utilities for object measures*

Description

- get_measures() computes object measures (area, perimeter, radius) by using either a known resolution (dpi) or an object with known measurements.
- plot_measures() draws the object measures given in an object to the current plot. The object identification ("id") is drawn by default.

Usage

```
get_measures(
  object,
  id = NULL,
  measure = NULL,
  dpi = NULL,
```

```
  verbose = TRUE,
  digits = 3
)

plot_measures(
  object,
  measure = "id",
  digits = 2,
  size = 0.9,
  col = "white",
  ...
)
```

Arguments

object	An object computed with count_objects() or leaf_area().
id	An object in the image to indicate a known value.
measure	For plot_measures(), a character string; for get_measures(), a two-sided formula, e.g., measure = area ~ 100 indicating the known value of object id. The right-hand side is the known value and the left-hand side can be one of the following. • area The known area of the object. • perimeter The known perimeter of the object. • radius_mean The known radius of the object. • radius_min The known minimum radius of the object. If the object is a square, then the radius_min of such object will be L/2 where L is the length of the square side. • radius_max The known maximum radius of the object. If the object is a square, then the radius_max of such object according to the Pythagorean theorem will be L x sqrt(2) / 2 where L is the length of the square side.
dpi	A known resolution of the image in DPI (dots per inch).
verbose	If FALSE, runs the code silently.
digits	The number of significant figures. Defaults to 2.
size	The size of the text. Defaults to 0.9.
col	The color of the text. Defaults to "white".
...	Further arguments passed on to graphics::text().

Value

- get_measures() returns a data frame with the object id and the measures. If measure is informed, the pixel values will be corrected by the value of the known object, given in the unit of the right-hand side of measure. If dpi is informed, then all the measures will be adjusted to the known dpi.
- plot_measures() returns a NULL object, drawing the text according to the x and y coordinates of the objects in object.

Examples

```
library(pliman)
img <- image_import(image_pliman("objects_300dpi.jpg"))
image_show(img)
# Image with four objects with a known resolution of 300 dpi
# Higher square: 10 x 10 cm
# Lower square: 5 x 5 cm
# Rectangle: 4 x 2 cm
# Circle: 3 cm in diameter

# Count the objects using the blue band to segment the image
results <-
   count_objects(img,
                 index = "B")
plot_measures(results, measure = "id")

# Get object measures by declaring the known resolution in dots per inch
(measures <- get_measures(results, dpi = 300))

# Calculated diagonal of the object 1
# 10 * sqrt(2) = 14.14

# Observed diagonal of the object 1
measures[1, "radius_max"] * 2


# Get object measures by declaring the known area of object 1
get_measures(results,
             id = 1,
             area ~ 100)
```

`utils_objects` *Utilities for working with image objects*

Description

- `object_id()` get the object identification in an image.
- `object_coord()` get the object coordinates and (optionally) draw a bounding rectangle around multiple objects in an image.
- `object_isolate()` isolates an object from an image.

Usage

```
object_coord(
  image,
  id = NULL,
```

```
  index = "NB",
  invert = FALSE,
  fill_hull = FALSE,
  edge = 2,
  extension = NULL,
  tolerance = NULL,
  object_size = "large",
  show_image = TRUE
)

object_isolate(image, id = NULL, ...)

object_id(image, ...)
```

Arguments

image	An image of class Image.
id	• For object_coord(), a vector (or scalar) of object id to compute the bounding rectangle. Object ids can be obtained with object_id(). Set id = "all" to compute the coordinates for all objects in the image. If id = NULL (default) a bounding rectangle is drawn including all the objects. • For object_isolate(), a scalar that identifies the object to be extracted.
index	The index to produce a binary image used to compute bounding rectangle coordinates. See image_binary() for more details.
invert	Inverts the binary image, if desired. Defaults to FALSE.
fill_hull	Fill holes in the objects? Defaults to FALSE.
edge	The number of pixels in the edge of the bounding rectangle. Defaults to 2.
extension, tolerance, object_size	Controls the watershed segmentation of objects in the image. See count_objects() for more details.
show_image	Shows the image with bounding rectangles? Defaults to TRUE.
...	• For object_isolate(), further arguments passed on to object_coord(). • For object_id(), further arguments passed on to count_objects().

Value

- object_id() An image of class "Image" containing the object's identification.
- object_coord() A list with the coordinates for the bounding rectangles. If id = "all" or a numeric vector, a list with a vector of coordinates is returned.
- object_isolate() An image of class "Image" containing the isolated object.

Examples

```
library(pliman)
img <- image_import(image_pliman("la_leaves.JPG"))
```

```
# Get the object's (leaves) identification
object_id(img)

# Get the coordinates and draw a bounding rectangle around leaves 1 and 3
object_coord(img, id = c(1, 3))

# Isolate leaf 3
isolated <- object_isolate(img, id = 3)
image_show(isolated)
```

`utils_transform` *Spatial transformations*

Description

Performs image rotation and reflection

- `image_autocrop()` Provides automatic image cropping.
- `image_dimension()` Gives the dimension (width and height) of an image.
- `image_rotate()` Rotates the image clockwise by the given angle.
- `image_horizontal()` Converts (if needed) an image to a horizontal image.
- `image_vertical()` Converts (if needed) an image to a vertical image.
- `image_hreflect()` Performs horizontal reflection of the `image`.
- `image_vreflect()` Performs vertical reflection of the `image`.
- `image_resize()` Resize the `image`. See more at `EBImage::resize()`.
- `image_contrast()` Improve contrast locally by performing adaptive histogram equalization. See more at `EBImage::clahe()`.
- `image_filter()` Performs median filtering in constant time. See more at `EBImage::medianFilter()`.
- `image_blur()` Performs blurring filter of images. See more at `EBImage::gblur()`.

Usage

```
image_autocrop(
  image,
  edge = 5,
  parallel = FALSE,
  workers = NULL,
  verbose = TRUE
)

image_dimension(image, parallel = FALSE, workers = NULL, verbose = TRUE)

image_rotate(
  image,
```

```
  angle,
  bg_col = "white",
  parallel = FALSE,
  workers = NULL,
  verbose = TRUE
)

image_horizontal(image, parallel = FALSE, workers = NULL, verbose = TRUE)

image_vertical(image, parallel = FALSE, workers = NULL, verbose = TRUE)

image_hreflect(image, parallel = FALSE, workers = NULL, verbose = TRUE)

image_vreflect(image, parallel = FALSE, workers = NULL, verbose = TRUE)

image_resize(
  image,
  rel_size = 100,
  width,
  height,
  parallel = FALSE,
  workers = NULL,
  verbose = TRUE
)

image_filter(
  image,
  size = 3,
  cache = 512,
  parallel = FALSE,
  workers = NULL,
  verbose = TRUE
)

image_blur(image, sigma = 3, parallel = FALSE, workers = NULL, verbose = TRUE)

image_contrast(image, parallel = FALSE, workers = NULL, verbose = TRUE)
```

Arguments

image	An image or a list of images of class Image.
edge	The number of pixels in the edge of the cropped image. If edge = 0 the image will be cropped to create a bounding rectangle (x and y coordinates) around the image objects.
parallel	Processes the images asynchronously (in parallel) in separate R sessions running in the background on the same machine. It may speed up the processing time when image is a list. The number of sections is set up to 90% of available cores.

workers	A positive numeric scalar or a function specifying the maximum number of parallel processes that can be active at the same time.
verbose	If TRUE (default) a summary is shown in the console.
angle	The rotation angle in degrees.
bg_col	Color used to fill the background pixels, defaults to "white".
rel_size	The relative size of the resized image. Defaults to 100. For example, setting rel_size = 50 to an image of width 1280 x 720, the new image will have a size of 640 x 360.
width, height	Width and height of the resized image. These arguments can be missing. In this case, the image is resized according to the relative size informed in rel_size.
size	The median filter radius (integer). Defaults to 3.
cache	The the L2 cache size of the system CPU in kB (integer). Defaults to 512.
sigma	A numeric denoting the standard deviation of the Gaussian filter used for blurring. Defaults to 3.

Value

A modified version of image depending on the function used.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

Examples

```
library(pliman)
img <- image_import(image_pliman("sev_leaf.jpg"))
image_show(img)
img <- image_resize(img, 50)
img1 <- image_rotate(img, 45)
img2 <- image_hreflect(img)
img3 <- image_vreflect(img)
img4 <- image_vertical(img)
image_combine(img1, img2, img3, img4)
```

Index

∗ **images**
    pliman_images, 24

cm_to_dpi *(*utils_dpi*)*, 28
cm_to_pixels *(*utils_dpi*)*, 28
count_lesions, 2
count_objects, 5
count_objects(), *33*, *35*

dpi_to_cm *(*utils_dpi*)*, 28

EBImage::clahe(), *36*
EBImage::gblur(), *36*
EBImage::medianFilter(), *36*
EBImage::resize(), *36*
EBImage::watershed(), *4*

file_dir *(*utils_file*)*, 29
file_extension *(*utils_file*)*, 29
file_name *(*utils_file*)*, 29

get_measures *(*utils_measures*)*, 32
graphics::text(), *33*

image_autocrop *(*utils_transform*)*, 36
image_binary, 9
image_binary(), *8*, *23*, *35*
image_blur *(*utils_transform*)*, 36
image_combine, 11
image_combine(), *25*
image_contrast *(*utils_transform*)*, 36
image_dimension *(*utils_transform*)*, 36
image_export *(*utils_image*)*, 31
image_filter *(*utils_transform*)*, 36
image_filter(), *7*, *25*
image_horizontal *(*utils_transform*)*, 36
image_hreflect *(*utils_transform*)*, 36
image_import *(*utils_image*)*, 31
image_index, 11
image_index(), *4*, *7*, *10*, *15*, *22*, *25*
image_palette, 14
image_pliman *(*utils_image*)*, 31
image_resize *(*utils_transform*)*, 36
image_rotate *(*utils_transform*)*, 36
image_segment, 15
image_show *(*utils_image*)*, 31
image_to_mat, 17
image_vertical *(*utils_transform*)*, 36
image_vreflect *(*utils_transform*)*, 36

leaf_area, 18
leaf_area(), *33*

manipulate_files *(*utils_file*)*, 29

object_coord *(*utils_objects*)*, 34
object_coord(), *35*
object_id *(*utils_objects*)*, 34
object_id(), *35*
object_isolate *(*utils_objects*)*, 34
objects_rgb, 20

pixels_to_cm *(*utils_dpi*)*, 28
pliman_images, 24
pliman_indexes *(*utils_file*)*, 29
plot.image_index *(*image_index*)*, 11
plot.objects_rgb *(*objects_rgb*)*, 20
plot_measures *(*utils_measures*)*, 32
prop_segmented, 25

symptomatic_area, 26

utils_dpi, 28
utils_file, 29
utils_image, 31
utils_measures, 32
utils_objects, 34
utils_transform, 36